Multi-Agent ML Stock Trading

Ainesh Chatterjee[1] **Andrew Zheng**[1] **Alec Luterman**[1] **Suhaib Matar**[1] **Brian Diarra**[1]

Abstract

This paper presents a comparative study of machine learning-based stock prediction approaches using a standardized dataset. It involves training diverse profit-maximizing agents to react to market fluctuations and evaluating their performance. Framed within the gymnasium library environment, the study preprocesses data to ensure uniformity and adopts an ensemble of agents to prevent overfitting. Techniques, in increasing complexity, such as Autoregressive Integrated Moving Average (ARIMA) and Long Short Term Memory (LSTM) models are examined. Profit comparison models such as Black–Scholes are also considered, alongside Multi DQN (Deep Q-Network) for reinforcement learning. Results showcase varying precision levels across models, with LSTM and Multi DQN demonstrating superior performance, and Multi DQN having the highest accuracy. This study provides insights into ML-based stock prediction and suggests avenues for enhancing predictive accuracy through model collaboration and hybrid control mechanisms.

1. Introduction

The Context
In the realm of machine learning-based stock prediction, understanding the effectiveness of different models is crucial for informed decision-making in financial markets. This study delves into a comparative analysis of various machine learning algorithms applied to stock prediction, aiming to assess their performance and efficacy in capturing market dynamics.

The Challenge
With the proliferation of machine learning techniques in financial forecasting, there arises a need to systematically compare and evaluate these approaches. The challenge lies in discerning which models are most adept at predicting stock movements amidst market uncertainties and fluctuations.

Data Limitations and Challenges
Despite the robustness of machine learning techniques, our study encounters challenges stemming from the limited scope of data utilized. By focusing solely on OHLCV (Open, High, Low, Close, Volume) data, our analysis may overlook crucial factors influencing stock price movements. Stock prices are influenced by a myriad of factors including earnings reports, investor sentiment, macroeconomic variables, and geopolitical events, among others. The absence of such qualitative data in our analysis poses a significant challenge in accurately predicting stock price fluctuations and assessing the true efficacy of machine learning models in real-world trading scenarios. This limitation underscores the need for further research integrating a broader range of data sources to enhance the predictive capabilities of machine learning algorithms in stock prediction.

Our Approach
To address this challenge, our study adopts a rigorous methodology involving the training of diverse profit-maximizing agents to respond to market changes. We seek to compare the performance of models such as Autoregressive Integrated Moving Average (ARIMA), Long Short Term Memory (LSTM), and Multi Deep Q-Network (DQN) in capturing stock price movements.

Significance of Benchmarking
Benchmarking against simpler agents such as Random, Heuristic, and Buy-and-Hold provides a baseline for evaluating the effectiveness of more complex machine learning models. This comparative analysis sheds light on the strengths and limitations of each approach, contributing to a deeper understanding of machine learning-based stock prediction.

Organization of the Paper
The paper is structured to provide a comprehensive overview of our research methodology, including the framing of the study within the gymnasium library environment and the processing of standardized stock datasets. We then present the results of our comparative analysis, followed by a discussion of implications and avenues for future research in machine learning-based stock prediction.

2. Environment

The environment simulates large stock trading platforms that provide key features such as providing market data,

reading actions, and returning profit. In the following section, we will describe the initial goal for the environment, its limitations, and the reformed environment.

2.1. Data

To allow agents to train sufficiently on raw data, the environment reads OHLCV (Open, High, Low, Close, Volume) data from 24 individual stocks between years 2002 and 2018, incrementing by 5 minute intervals.

2.2. Initial Environment

In order to create a framework similar to environments used in common Reinforcement Learning models, the initial goal of the market simulator is to utilize Open AI's Gymnasium library. The environment processes the data by applying data-type transformations to ensure uniform features, and initializes a list of dictionaries for the stock name and corresponding data-frame, which will serve as the general market. Using Multi-DQN (Salvatore Carta, 2020) as a basic reference, the agent will be allowed to commit a step (action) in the market at the current observation. The function receives a list of actions (buy, sell, or hold) that correspond to the number of stocks. Depending on the action, the function returns a list of positive or negative rewards calculated by the percent change from open to close between an interval for each stock. For example, if the agent provides a buy action and the open-close change is positive, the reward will likewise be positive. However, if the agent provides a sell action in the same scenario, the reward will be the negative percent of the open-close change.

2.3. Initial Environment Limitations

As the agents in *Section 3* were being developed, the environment seemed to become less necessary because each model only required the market data to train. Unlike Reinforcement Learning agents that use gymnasium for game-like situations, the agent cannot return to a previously visited observation, and therefore, only needs an incrementing index to iterate. Additionally, due to the extremely short intervals for each data point, rewards experienced too little change to return meaningful profit. Our initial goal of having an independent environment that acts as a playground for agent progress observation is unnecessarily complicated at least for market simulation.

2.4. Reformed "Environment"

Currently, each agent is accompanied with a quasi environment, which is simply a list of data-frames for each stock, instead of wrapping it within an environment class. Not only does this new format simplify unnecessary code, but also allows for direct access to data-frame indices that benefits training and testing. As a result, agents are trained to predict future market values and assign actions dependent on those values instead of relying on an open-close percent reward function.

3. Agents

We chose the following agents, each of increasing complexity, in order to compare the results of different models.

3.1. ARIMA

The Autoregressive Integrated Moving Average (ARIMA) model, is a forecasting method that captures patterns and dependencies within sequential data (Vijay Kotu, 2019). ARIMA consists of 3 main components: autoregression, differencing, and moving average.

The autoregressive component represents the relationship between an observation and a given number of lagged observations. This essentially models the dependence of the current value in a time series on its past values.

The differencing component transforms the time series into a stationary series by removing trends or seasonality. This is required because ARIMA models are sensitive to the characteristics of the time series data they are applied to, specifically stationarity. Stationarity implies that properties of the time series such as mean, variance and autocorrelation, remain constant over time.

Before fitting an ARIMA model, it's essential to check for stationarity, which can be done through statistical tests such as the Augmented Dickey-Fuller (ADF) test. If the data is non-stationary, it needs to be transformed in order to achieve stationarity.

The moving average component captures the relationship between an observation and a residual error from a moving average model applied by lagged observations.

ARIMA models can effectively predict future values in a time series by iteratively adjusting the parameters of each of these components

3.2. LSTM Agent

Recurrent Neural Networks (RNN) are a popular type of neural network but are stricken by the vanishing gradient problem, where in deep neural networks, the weights of the deeper layers are changed only marginally when training. In effect very little to no learning is done. One resolution to this is a Long Short-Term Memory (LSTM) network. LSTMs are a type of RNN that include the addition of three types of gates: input gate, forget gate, and output gate. These gates control the information flow between different layers of the network.(lst) In the context of the stock market and

finance, LSTMS are useful because of their ability to handle sequential data and time series.(Fjellström, 2022)

(CodeTrading, 2022)For this project, we trained a different LSTM model for each stock in order to accurately match the characteristics of each stock and predict the next closing stock price. We used Keras to create the LSTM models. For every stock, the training loss for the last epoch was less than 1e-4.

In the version of the LSTM agent that worked with the environment, a wrapper/helper class was created in order to work with the environment's respective step() function. This class would load all of the trained models, and prepare all of the data for testing/evaluation. The more important aspect of this class was the action() function which would go through each agent, make the price prediction for the current observation index, and make a decision on whether to buy sell or hold. This decision is based on a simple heuristic comparing the difference in predicted price and the previous price to a threshold. Since the LSTM agent does not use the environment reward, the only benefit to using the environment is to make observations one time index at a time.

The final version of the LSTM agent simplified this by not requiring the use of a helper method since it was not intended to work with a gym environment. Because of this, we can simply iterate over the stocks and use the keras predict() function to make all predictions for the test data at once. In addition, these price predictions are fed into the profit evaluation code in order to make more accurate decisions on what to do with each stock at each time step.

3.3. MultiDQN Agent

The Multi-DQN model, as proposed by Carta et al. (Salvatore Carta, 2020), presents an innovative approach to stock market forecasting through an ensemble of reinforcement learning agents. This model employs multiple instances of Deep Q-Networks (DQNs), trained on the same dataset but initiated at different epochs, to capitalize on diverse learning experiences and perspectives.

3.3.1. Motivation and Theoretical Foundation

The stock market is characterized by its complex and chaotic behavior, influenced by numerous unpredictable factors such as economic changes, political events, and market sentiment. Traditional predictive models often struggle with overfitting and lack the adaptability required for the dynamic nature of financial markets. Multi-DQN addresses these challenges by integrating multiple learning agents, each providing a unique strategic output, which collectively contribute to a more robust prediction system.

3.3.2. Mechanics of Multi-DQN

In the Multi-DQN framework, each agent operates independently, utilizing a standard Q-learning algorithm where the reinforcement learning agent interacts with an environment to maximize a cumulative reward. The distinctiveness of each agent's learning path is ensured by differing initialization points, promoting diversity in the learned policies and reducing the risk of collective overfitting.

3.3.3. Collaborative Decision-Making

The ensemble approach in Multi-DQN leverages the diversity among the agents to make more informed and balanced trading decisions. By employing a majority voting scheme or calculating the average of decisions across all agents, the system can mitigate the impact of potentially erroneous predictions from any single model. This collaboration facilitates a consensus-driven approach to trading decisions, enhancing the reliability of the predictions.

3.3.4. Advantages Over Single Model Approaches

The Multi-DQN methodology offers several advantages over traditional single-model forecasting techniques:

- **Enhanced Generalization:** The ensemble method helps in generalizing better across different market conditions, reducing the likelihood of overfitting to specific historical data trends.
- **Risk Diversification:** Multiple agents contribute to a risk diversification strategy, as the failure of one agent does not necessarily compromise the overall system's performance.
- **Adaptability:** The system can adapt more effectively to market volatility due to the collective intelligence and varied experiences of the ensemble agents.

3.3.5. Current Limitations and Future Work

Despite its promising approach, the Multi-DQN system is currently not without issues. As highlighted in ongoing tests, the system's complexity and the requirement for extensive computational resources pose significant challenges. Future enhancements could include optimizing the learning efficiency of individual agents and refining the ensemble methods to improve decision accuracy further.

Note: The code implementations and methodologies discussed are still not fully functional. This highlights an ongoing area of research that promises to evolve with further investigation and refinement.

4. Results

Please refer to the *analysis.ipynb* file under the analysis folder in the $final$ branch of the GitHub repository.

5. Future Work

5.1. ARIMA

Although ARIMA models demonstrated strong performance with a rolling forecasting structure, the difference between actual and predicted market values gradually increase in size as the observation progresses, despite accurately predicted patterns and spike trends. We plan on identifying and resolving this issue, unless inherent to internal ARIMA machine learning.

5.2. Mixture of Experts (MoE)

Since our initial goal was to generate a multi-agent model, we plan to use our existing agents to create a MoE. This machine learning architecture uses a few "expert" models to predict a singular action, and relies on a gating network to assign weights to each experts' predictions during an observation. This model allows for better generalization and reduces the possibility for over-fitting, especially for the uncertainty and high volatility of some stock markets.

6. Contributions

6.1. Ainesh Chatterjee

1. ARIMA Agent
2. Modifying LSTM Agent for final version
3. Options profit calculation logic for ARIMA + LSTM
4. MultiDQN code adaptation attempt
5. MultiDQN writeup subsection, i.e. section 3.3

6.2. Andrew Zheng

1. Data Processing
2. Environment (markEnv.py)
3. Initial Agents Result Analysis Notebook (ie. Precision, Net Profit)
4. Environment and Future Work Section Writeup, ie. section 2 and 5

6.3. Alec Luterman

1. LSTM Agent
2. All files supplementary to LSTM such as wrapper class and training files (MarkertandLSTM branch in GitHub)
3. Stock Forecasting and Profitability Analysis
4. Writeup for LSTM agent

6.4. Suhaib Matar

1. Environment and agent integration research
2. Environment setup
3. Writeup for abstract and introduction
4. Assistance in the literature review

6.5. Brian Diarra

1. Benchmark Agents
2. Stock Forecasting and Profitability Analysis
3. Writeup for ARIMA model

References

Long short-term memory (lstm). URL https://developer.nvidia.com/discover/lstm.

CodeTrading. Recurrent neural networks — lstm price movement predictions for trading algorithms. 2022.

Fjellström, C. Long short-term memory neural network for financial time series. 2022. URL https://arxiv.org/pdf/2201.08218.

Salvatore Carta, Anselmo Ferreira, A. S. P. D. R. R. A. S. Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting. 2020. URL https://www.sciencedirect.com/science/article/pii/S0957417420306321.

Vijay Kotu, B. D. Chapter 12 - time series forecasting. *Data Science (Second Edition)*, 2019. URL https://doi.org/10.1016/B978-0-12-814761-0.00012-5.

A. You *can* have an appendix here.

You can have as much text here as you want. The main body must be at most 8 pages long. For the final version, one more page can be added. If you want, you can use an appendix like this one, even using the one-column format.